Penghindaran Rintangan untuk Robot TurtleBot3 dengan Menggunakan Algoritma *Greedy* dan Sensor *Laser Scan*

Rizky Abdillah Rasyid - 13521109
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13521109@std.stei.itb.ac.id

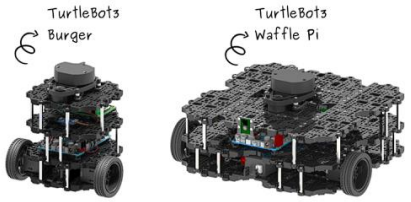
***Abstrak*—Robotika merupakan bidang yang berkembang pesat dalam beberapa waktu terakhir. Salah satu aspek penting dalam robotika adalah kemampuan dalam berinteraksi dengan lingkungannya. Kemampuan tersebut salah satunya penghindar rintangan. Penghindar rintangan merupakan aspek penting dalam pengembangan robotika. Dalam pengembangan aspek tersebut diperlukan kombinasi kemampuan perangkat keras dan kemampuan perangkat lunak. Laser scan merupakan sensor yang dapat mendeteksi lingkungan mirip seperti LiDAR. Algortima *Greedy* dapat digunakan sebagai salah satu pilihan algortima yang dapat menentukan lintasan agar robot bergerak dengan menghindari rintangan. Kombinasi sensor *laser scan* dengan algortima *greedy* dapat menjadi solusi untuk mendapatkan kemampuan penghindar rintangan pada robot.**

Keywords—robot, algortima, greedy, penghindar-rintangan, obstacle-avoidance, sensor, laser-scan

I. PENDAHULUAN

Robotika merupakan salah satu bidang yang berkembang pesat dalam beberapa waktu terakhir. Dalam teknologi robotika, rekayasa komputasi berperan besar dalam membentuk wahana robotika yang dalam menyelesaikan tugasnya. Dalam mengembangkan wahana robotika, salah satu aspek penting adalah kemampuan robot untuk berinteraksi dengan lingkungan sekitarnya. Dalam konteks ini, kemampuan navigasi dan menghindari rintangan menjadi perhatian utama. Navigasi pada robotika merupakan kemampuan robot dalam bergerak secara efektif dan efisien untuk mencapai titik tujuan. Penghindar rintangan (*obstacle avoidance*) merupakan salah satu aspek penting dalam navigasi pada robot. Penghindar rintangan dikembangkan dengan mengkombinasikan kemampuan perangkat keras sensor yang mengubah kondisi lingkungan sebenarnya menjadi data dengan kemampuan perangkat lunak berupa program yang melakukan rekayasa dari data yang diterima menjadi suatu informasi atau perintah terusan sehingga suatu robot dapat menanggapi suatu kondisi pada lingkungan.

Robot Turtlebot3 merupakan salah satu model robot yang populer digunakan dalam penelitian dan pengembangan bidang robotika yang menggunakan framework Robot Operating System. Model ini dikembangkan oleh ROBOTIS dan merupakan model *open-source* yang mudah digunakan. Model ini dilengkapi oleh kamera, sensor jarak, sensor LaserScan dan odometry. Sensor LaserScan merupakan representasi LiDAR dalam lingkungan simulasi, LiDAR merupakan sensor yang menggunakan sinar laser untuk menghitung jarak dan mendeteksi objek di sekitarnya. Cara kerja LiDAR dengan menembakkan laser kepada lingkungan dan jarak yang dihasilkan berasal dari waktu yang dibutuhkan laser memantul kembali ke laser. Selain mengembalikan data jarak, LiDAR juga mengambalikan data berupa *point cloud* yang menggambarkan bentuk permukaan objek yang dideteksinya. Untuk mengetahui posisi letak robot, digunakan *odometry* yang dapat mengestimasi posisi pergerakan dan posisi dengan acuan titik awal robot beroperasi merupakan titik *origin*.



Gambar 1. 1 Model Turtlebot3 *"Burger"* dan *"Waffle"*
Sumber : https://emanual.robotis.com/assets/images/platform/turtlebot3/overview/turtlebot3_with_logo.png

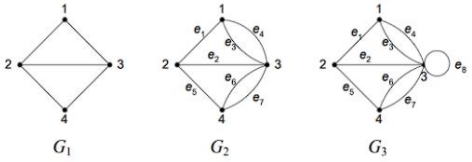
Kemampuan perangkat lunak (program) menjadi hal penting dalam pengembangan robot dalam aspek penghindar rintangan. Program yang diaplikasikan harus mampu memanipulasi data dari sensor dengan struktur data tertentu dan mengolahnya dengan algoritma tertentu untuk mendapatkan perintah yang optimal. Optimal dalam konteks ini adalah perintah yang menggerakkan robot mencapat titik tujuan dengan jalur yang minimum dan berhasil menghindari semua rintangan. Salah satu pilihan algoritma yang dapat diterapkan adalah algoritma *greedy*. Algoritma *greedy* merupakan metode yang paling populer untuk memecahkan masalah optimasi yang bekerja dengan mengambil pilihan terbaik yang dapat diperoleh pada saat itu tanpa memperhatikan kosekuensi kedepan[1]. Dalam makalah ini, penuli berfokus pada pembahasan algortima greedy dalam pemecahan masalah robot penghindar rintangan. Simulasi akan dilakukan tanpa menggunakan program simulasi, tetapi akan dilakukan berdasarkan tahapan-tahapan kejadian. Selain itu, pada makalah ini, penulis menggunakan struktur data

graf untuk mengubah data sensor laser scan menjadi struktur data yang dapat diproses pada program.

II. LANDASAN TEORI

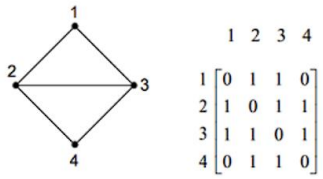
A. Graf

Graf adalah representasi dari kumpulan objek-objek disktrit dan hubungan antara objek-objek tersebut. Graf didefinisikan sebagai tuple G = (V, E), dengan V adalah himpunan tak kosong dari simpul-simpul (vertices) dan E adalah himpunan sisi (edges) yang menjadi penhubung antara dua simpul.



Gambar 2. 1 Graf
Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf, diakses pada 12/12/2022

Untuk melakukan rekayasa komputasi pada graf, Graf direpresentasikan dengan matriks ketetanggaan (Adjacency Matrix), indeks baris dalam kolom pada matriks ini merepresentasikan simpul-simpul pada graf. Elemen pada matriks ini bertipe boolean (0/1) yang mengindikasi apakah antara dua simpul (indeks baris dan kolom) terdapat sisi yang menghubungkannya. Pada pengaplikasiannya, indeks baris dan kolom dapat dapat diubah menjadi representasi dari objek nyata lain.



Gambar 2. 2 Graf Ketetanggaan
Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian2.pdf, diakses pada 12/12/2022

First, confirm that you have the correct template for your paper size. This template has been tailored for output on the A4 paper size. If you are using US letter-sized paper, please close this file and download the file "MSW_USltr_format".

B. Algoritma Greedy

Algortima *greedy* merupakan paradigma pengambilan keputusan sederhana yang bersifat serakah (*greedy*). Algortima ini biasa digunakan untuk memecahkan masalah optimasi atau mencari solusi optimasi, hanya ada dua persoalan optimasi, yaitu maskimasi dan minimasi. Minimasi adalah pencarian solusi paling minimum. Sedangkan, maksimasi adalah pencarian solusi paling maskimum. Algoritma ini melakukan pemecahan maslaah secara bertahap (*step by step*), setiap langkahnya perlu mengambil pilihan terbaik yang diperoleh pada saat itu tanpa memperhatikan kosekuensi kedepannya, prinsip pada algortima *greedy* ini biasa disebut *"take what you can get now"*. Setiap pengambilan pilihan kita berharap dengan memilih pilihan tersebut bersifat optimum lokal pada setiap langkahnya dan akan berakhir dengan optimum global.

Dalam memecahkan masalah dengan algoritma *greedy*, digunakaan elemen-elemen berikut:

1. Himpunan kandidat (C)
   Himpunan kandidat berisi kandidat yang akan dipilih pada setiap langkahnya. Kandidat dalam konteks ini adalah pilihan aksi atau nilai yang tersedia.

2. Himpunan Solusi (S)
   Himpunan solusi adalah himpunan yang berisis kandidat yang sudah terpilih karena memenuhi solusi.

3. Fungsi Solusi
   Fungsi solusi adalah fungsi yang menentukan apakah himpunan kandidat yang diberikan sudah memeberikan solusi.

4. Fungsi Seleksi (selection function)
   Fungsi seleksi adalah fungsi yang memilih kandidat berdasarkan heuristik yang ditentukan.

5. Fungsi Kelayakan (feasible)
   Fungsi kelayakan adalah fungsi yang memeriksa kandidat yang dipilih apakah bisa dimasukkan ke dalam himpunan solusi.

6. Fungsi Objektif
   Fungsi objektif adalah fungsi yang meminimumkan atau memaksimalkan.

Algortima *greedy* mungkin tidak selalu berhasil memberikan solusi yang optimal, namum sub-optimal. Artinya mendekati nilai yang optimal. Heuristik dalam konteks algoritma adalah aturan atau strategi yang digunakan untuk membuat keputusan pendekatan berdasarkan pengetahuan dan pengalaman sebelumnya, meskipun tanpa jaminan solusi yang optimal [2]. Dalam penentuannya, heuristik ditentukan sesuai dengan kebutuhan pemecahan masalah yang dihadapi.

C. Robot Operating System

Robot Operating System (ROS) adalah kerangka kerja (*freamework*) *open-source* yang populer digunakan dalam penelitian dan pengembangan perangkat lunak dalam robotika. Karena bersifat *open-source*, ROS memiliki komunitas yang besar sehingga banyak contributor yang menyediakan pustaka dan alat-alat yang dapat digunakan untuk pengembang atau peneliti mengembangkan robot secara efisien [3].

Dalam pengembangan preangkat lunak dengan ROS, pengembangan dibagi menjadi bebrapa node yang berjalan independen (seperti thread) yang dapat berkomunikasi melalui pesan. Pada ROS, pesan ini biasa disebut *rosmessage* dan jalur atau *pipeline* yang digunakan untuk berkirim pesan disebut *rostopic*. Kerangka kerja ini memiliki lingkungan simulasi bernama Gazebo yang dapat melakukan pengujian robot sebelum diterapkan pada komputer robot sebenarnya. Gazebo

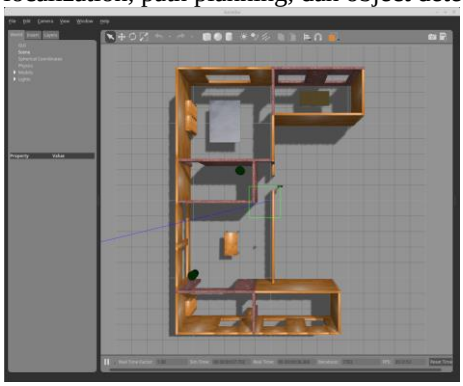
dapat menjalankan model robot yang dapat dibuat sendiri oleh pengembang atau mendapatkan dari penyedia *open-source* model.

D. *Turtlebot3*

Turtlebot3 merupakan platform robotika yang dibangun dalam kerangka kerja ROS yang dikembangkan oleh ROBOTIS dan bertujuan untuk pengembangan dan eksperimen bidang robotika.

Gambar 2. 3 Model turtlebot3 pada Gazebo
Sumber :
https://emanual.robotis.com/assets/images/platform/turtlebot3/simulation/turtlebot3_empty_world.png

Turtlebot3 dilengkapi dengan komponen yang bersifat modular untuk mendukung pengembangan dan eksperimen secara efisien. Komponen yang dimaksud seperti: LiDAR, Actuator Pengerak roda dan Odometry. Pada aspek pengembangan perangkat lunak pada robot, turtlebot juga menyediakan *package* yang dapat digunakan untuk melakukan mapping, localization, path planning, dan object detection.



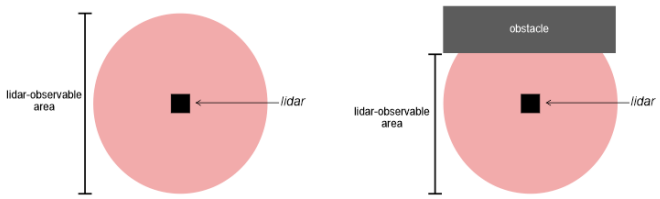
Gambar 2. 4 Model lingkungan pengujian turtlebot3
Sumber :
https://emanual.robotis.com/assets/images/platform/turtlebot3/simulation/turtlebot3_house.png

Turtlebot3 memiliki model robot beserta model lingkungan pengujian yang dapat disimulasikan pada lingkungan simulasi Gazebo, sehingga pengembang tidak perlu melakukan eksperimen paa robot fisik secara langsung untuk pengujian. Agar program dapat mengakses data sensor atau mengirimkan perintah pada robot, perlu dilakukan langganan (*subscribe*) atau publikasi (*publish*) pada rostopic yang disediakan oleh turtlebot3. Berikut adalah rostopic yang tersedia:

1. */cmd_vel* (Tipe Pesan: *geometry_msgs/Twist*) Rostopic ini digunakan untuk mengirim perintah ke TurtleBot3 Burger untuk mengatur kecepatan dan arah gerakan robot. Pesan yang dikirim melalui topik ini memuat kecepatan linear dan sudut angular yang diinginkan.
2. */scan* (Tipe Pesan: *sensor_msgs/LaserScan*) Rostopic ini digunakan untuk menerima data pemindaian lidar dari sensor LaserScan pada TurtleBot3 Burger. Data pemindaian lidar berisi informasi tentang jarak dan intensitas pantulan laser dari objek di sekitar robot.
3. */odom* (Tipe Pesan: *nav_msgs/Odometry*) Rostopic ini digunakan untuk menerima data odometri dari TurtleBot3 Burger. Data odometri memberikan informasi tentang pergerakan dan perkiraan posisi robot berdasarkan sensor-sensor yang terpasang di robot.
4. */tf* (Tipe Pesan: *tf2_msgs/TFMessage*) Rostopic ini digunakan untuk mentransmisikan transformasi koordinat (transform frames) antara berbagai frame referensi yang ada dalam TurtleBot3 Burger. Informasi transformasi koordinat ini diperlukan untuk menghubungkan dan memetakan posisi dan orientasi antara berbagai komponen robot, seperti sensor, pemutar motor, dan basis robot.

E. *Laser Scan*

Laser scan merupakan representasi LiDAR dalam model simulasi turtlebot3. Sama seperti LiDAR, *laser scan* melakukan pendeteksian objek dan pengukuran jarak disekitarnya yang bekerja dengan memancarkan laser ke lingkungan dan mengukur waktu yang diperlukan Kembali memantul dari objek sekitar.



Gambar 2. 5 Bentuk area deteksi laser-scan
Sumber :
Dokumentasi pribadi

Data yang didapat dari laser scan berbentuk *message* bernama *sensor_msgs/LaserScan*. Tipe data tersebut memiliki struktur sebagai berikut:

1. Header:
   a. Seq: Nomor urutan pesan dalam timestamp [4].
   b. Stamp: Timestamp ketika data pemindaian lidar diambil [4].
   c. Frame_ID: Nama frame referensi yang digunakan [4].
2. Angle Min dan Angle Max:
   a. Angle Min: Sudut awal pemindaian lidar dalam radian [4].
   b. Angle Max: Sudut akhir pemindaian lidar dalam radian [4].
3. Angle Increment:
   a. Nilai inkremental sudut antara setiap sampel pemindaian lidar dalam radian [4].

4. Time Increment:
    a. Nilai inkremental waktu antara setiap sampel pemindaian lidar dalam detik [4].
5. Scan Time:
    a. Durasi total pemindaian lidar dalam detik [4].
6. Range Min dan Range Max:
    a. Range Min: Jarak minimum yang dapat diukur oleh lidar dalam satuan meter [4].
    b. Range Max: Jarak maksimum yang dapat diukur oleh lidar dalam satuan meter [4].
7. Ranges:
    a. Array nilai jarak terukur pada setiap sudut pemindaian lidar. Nilai ini mewakili jarak dari sensor ke titik-titik di sekitarnya [4].
8. Intensities (opsional):
    a. Array nilai intensitas pantulan lidar pada setiap sudut pemindaian. Nilai ini mewakili tingkat kecerahan atau intensitas pantulan pada titik-titik tersebut. Informasi intensitas tidak selalu tersedia dalam setiap pemindaian lidar [4].

F. Odometry

Odometry berperan dalam mengestimasi posisi dan gerakan dari robot berdasarkan data dari sensor lain. Odometry menentukan posisi dengan menghitung perubahan posisi dan orientasi setiap waktu. Odometry biasnaya akan menjadikan titik awal alat diaktifkan sebagai titik origin (0,0) sehingga frame (kerangka) lingkungan robot berbeda dengan lingkungan asli, sehingga perlu dilakukan konversi dari frame odometry ke frame lingkungan asli agar tidak menyebabkan error yang besar ketika robot bergerak. Data yang didapat dari odometry berbentuk *message* bernama *nav_msgs/Odometry*. Tipe data tersebut memiliki struktur sebagai berikut:

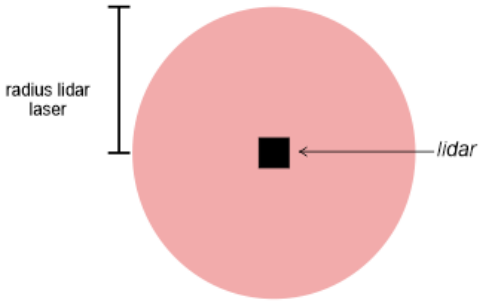
1. Header:
    a. Seq: Nomor urutan pesan dalam timestamp.
    b. Stamp: Timestamp ketika data odometri diambil.
    c. Frame_ID: Nama frame referensi yang digunakan.
2. Child_Frame_ID:
    a. Nama frame referensi yang melekat pada robot, seperti base_link atau base_footprint.
3. Pose:
    a. Pose (posisi dan orientasi) perkiraan robot dalam sistem koordinat global.
        i. Pose.Position:
            1. X: Komponen X posisi perkiraan robot dalam meter.
            2. Y: Komponen Y posisi perkiraan robot dalam meter.
            3. Z: Komponen Z posisi perkiraan robot dalam meter.
        ii. Pose.Orientation:
            1. X: Komponen X orientasi perkiraan robot.
            2. Y: Komponen Y orientasi perkiraan robot.
            3. Z: Komponen Z orientasi perkiraan robot.
            4. W: Komponen W orientasi perkiraan robot.
    b. Twist, Kecepatan dan perubahan sudut perkiraan robot.
        i. Twist.Linear:
            1. X: Kecepatan linear perkiraan robot dalam sumbu X.
            2. Y: Kecepatan linear perkiraan robot dalam sumbu Y.
            3. Z: Kecepatan linear perkiraan robot dalam sumbu Z.
        ii. Twist.Angular:
            1. X: Kecepatan sudut perkiraan robot dalam sumbu X.
            2. Y: Kecepatan sudut perkiraan robot dalam sumbu Y.
            3. Z: Kecepatan sudut perkiraan robot dalam sumbu Z.

III. METODE

Metode dalam pengembangan ini dibagi menjadi 2 bagian yaitu, pra-proses dan proses *greedy*.

A. Pra-Proses

Pra-proses merupakan tahap pemrosesan data dari sensor menjadi tipe data yang dapat dimengerti dan diproses oleh program yang menerapkan algoritma *greedy*. Pada makalah ini akan diatur radius laser (Range Max) berukuran 1.0-meter, artinya objek yang berada pada jarak lebih dari 1.0-meter dari robot tidak akan terdeteksi.



Gambar 3. 1 Ilustrasi radius laser

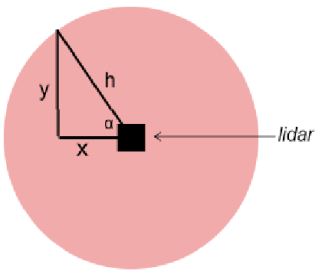
Pre-proses dilakukan dengan mengubah data dari tipe data sensor_msgs/LaserScan menjadi graf ketetanggaan. Berikut adalah rancangan kode dalam bentuk (pseudocode):

```
function preprocess(scan_msg):
    ranges = scan_msg.ranges
    angle_min = scan_msg.angle_min
    angle_increment = scan_msg.angle_increment

    graph = createGraph(21, 21)

    for i = 0 to length(ranges) - 1:
        angle = angle_min + i * angle_increment
        x = ranges[i] * cos(angle)
        y = ranges[i] * sin(angle)

        gx = 10 + (round(10*x)/10) * 10
        gy = 10 + (round(10*y)/10) * 10

        if gx is within graph bounds and gy is within
           graph bounds:
            graph[gx][gy] = 1
```

Gambar 3. 2 Pseudocode tahap pra-proses
Sumber :
Dokumentasi pribadi

Pada kode ini, alasan dibentuk graf berukuran 21 x 21 karena radius yang digunakan dalam pendeteksian laser berukuran 1 meter. Artinya laser hanya dapat mengambil data pada posisi axis maksimal adalah 1 meter, minimal -1 meter dan posisi ordinat maksimal adalah 1 meter, minimal -1 meter dengan acuan titik *origin* (0,0) adalah posisi robot pada saat itu. Untuk menghitung nilai x dan y, digunakan formula trigonometri sederhana,

$$x = h \times \cos\alpha$$
$$y = h \times \sin\alpha$$



Gambar 3. 3 Ilustrasi geometry pada laser
Sumber :
Dokumetasi Pribadi

Karena nilai x dan y dapat bersifat negatif, maka perlu dikonversi menjadi indeks yang dapat dimasukkan ke dalam graf. Konversi dilakukan dengan rumus berikut,

$$i = 10y + 10$$
$$j = 10x + 10$$

$$(x, y) = [-1.0, -0.9, -0.8, \dots, 0.8, 0.9, 1.0]$$
$$(j, i) = [0, 1, 2, 3, 4, \dots, 18, 19, 20]$$

Nilai x dan y akan didapat ketika pada suatu titik terdapat obstacle, sehingga ketika i dan j memiliki nilai dengan batas yang ditetapkan yaitu $0 \leq i \leq 20$ dan $0 \leq j \leq 20$, graf akan bernilai 1 pada graf[i][j]. Dari hasil konversi tersebut akan didapat matriks ketetanggaan yang merepresentasikan graf. Setelah itu graf akan diproses dengan algortima *greedy*.

B. Proses Greedy

Pada tahap ini akan dilakukan pemrosesan algortima *greedy*. Elemen pada algoritma *greedy* ini sebagai berikut,

1. Himpunan kandidat: semua titik koordinat pada graf.
2. Himpunan solusi: titik koordinat pada yang mungkin sebagai tujuan gerak.
3. Fungsi Solusi: memeriksa apakah pada pada titik koordinat tersebut bernilai 0.
4. Fungsi Seleksi: memilih titik koordinat mulai dari titik yang terdekat dengan goal.
5. Fungsi Kelayakan: memeriksa apakah jika ditarik garis lurus ke titik koordinat kadidat tersebut, tidak terdapat titik yang bernilai 1.
6. Fungsi objektif: jarak estimasi yang digunakan minimum. Jarak estimasi didapatkan dengan persamaan,

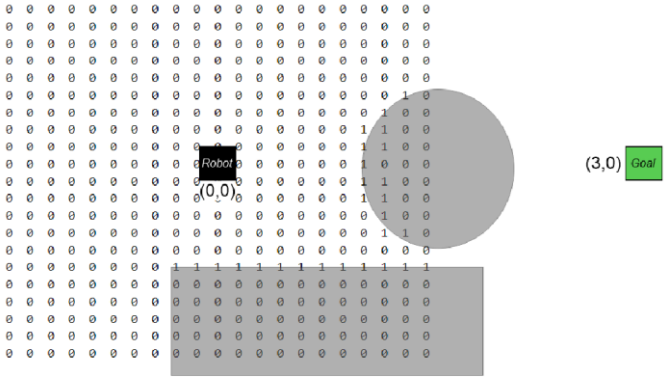
$$f = g(i, j) + h(i, j)$$

$f = jarak\ estimasi$
$g = jarakgaris\ lurus\ robot\ dari\ titik\ koordinat\ kandidat$
$h = jarak\ garis\ lurus\ dari\ titik\ tujuan\ dengan\ titik\ koordinat\ kandidat$

Setelah ditentukan titik tujuan, titik akan diteruskan ke fungsi penggerak wahana sehingga wahana bergerak dengan arah yang sama dengan titik tujuan. Pada fungsi penggerak, program akan melakukan *sleep* selama 10 detik. Setelah itu wahana akan diberhentikan dan akan dipanggil lagi proses *greedy* untuk menentukan tujuan baru hingga mencapai titik tujuan, semua node akan berhenti bekerja.

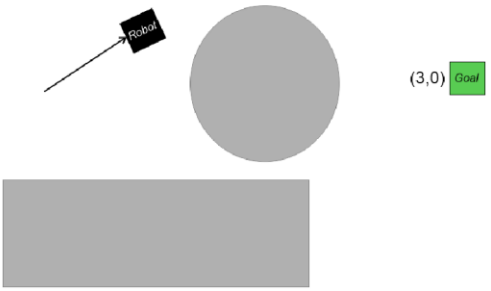
IV. HASIL

Proses dilakukan dengan kondisi berikut,



Gambar 4. 1 Ilustrasi lingkungan simulasi
Sumber :
Dokumentasi Pribadi

Pada kondisi ini akan dipilih titik bernilai 0, pada graf, titik pada i = 14 dengan j = 20 tidak layak berdasarkan fungsi kelayakan sehingga tidak layak dimasukkan ke dalam himpunan solusi. Dari himpunan solusi yang ada dengan fungsi objektif maka dipilih titik i = 4 dan j = 20 dengan nilai x=1 dan j=-0.4. maka robot akan bergerak ke arah tersebut.



Gambar 4. 2 Ilustrasi ketika bergerak menghindar objek rintangan
Sumber :
Dokumentasi pribadi

Setelah perintah bergerak dipanggil dan akan sleep selama 10 detik. Dan proses diulangi hingga robot telah mencapai titik tujuan (*goal*).

V. KESIMPULAN

Algoritma *Greedy* berhasil diterapkan pada wahana penghindar rintangan dengan data hasil pendeteksi objek dengan *laser scan*. Algoritma yang diterapkan tidak menjanjikan solusi yang optimal, hal ini bergantung ketelitian pada jumlah titik yang dievaluasi dan kesalahan pengukuran pada sensor.

VI. SARAN

Saran untuk penulis adalah melakukan pengujian dilingkungan simulasi untuk lebih meyakinkan hasil pengujian dan dapat mengetahui anomali atau *error* yang terjadi pada pengujian dengan wahana.

UCAPAN TERIMA KASIH

Puji dan syukur penulis ucapkan kepada Tuhan Yang Maha Esa karena atas izinnya penulis dapat menyelesaikan makalah ini. Penulis juga mengucapkan terima kasih kepada seluruh pihak yang telah membantu penulis dalam menyelesaikan makalah ini, antara lain:

1. Dr. Ir. Rinaldi Munir, M.T., Dr. Nur Ulfa Maulidevi, S.T, M.Sc, dan Ir. Rila Mandala, M.Eng., Ph.D, sebagai dosen pangampu dan penyusun materi mata kuliah IF2211 Strategi Algoritma.
2. Seluruh pihak yang telah membuat *open-source tools* yang digunakan untuk menyususn makalah ini.

REFERENCES

[1] R. Munir, "Algortima Greedy," 2023. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf. [Accessed 22 May 2023].

[2] T. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, Introduction to Algorithms 3rd ed., Boston: MIT Press, 2009.

[3] M. Quigley, B. Gerkey and W. D. Smart, "ROS: an open-source Robot Operating System," in *ICRA workshop on open source software*, Kobe, 2009.

[4] ROS Community, "sensor_msgs/LaserScan documentation," 28 February 2022. [Online]. Available: http://docs.ros.org/en/melodic/api/sensor_msgs/html/msg/LaserScan.html. [Accessed 22 May 2023].

[5] Ros, "ROS: Home," Open Robotics, 2021. [Online]. Available: https://www.ros.org/. [Accessed 12 Desember 2022].

[6] A. Koubaa, Robot Operating System (ROS) The Complete Reference (Vol 1), Riyadh: Springer, 2016.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 22 Mei 2023

Rizky Abdillah Rasyid 13521109